**COS-511: Learning Theory** **Spring 2017**

# Lecture 7: General Learning Problems, and Computational Learning Theory

*Lecturer: Roi Livni*

**Disclaimer**: *These notes have not been subjected to the usual scrutiny reserved for formal publications. They may be distributed outside this class only with the permission of the Instructor.*

## 7.1 Extending the PAC Model beyond Binary Classification

We next, extend the notion of learnability to accomodate a larger variety of learning problems. Here are some examples to learning problems that are not capture by the standard model
**Multiclass Classification** We do not allways want to return a binary number as our output. For example in document classification, given a document we might want to return a topic (e.g. News, Sports, Biology etc.). Our learning algorithm will receive as input documents, and will return a topic which belongs to a label set $\mathcal{Y} = \{\text{news}, \text{sports}, \text{biology}\}$. The label set might also be infinite $\mathcal{Y} = \mathbb{N}$.
**Regression** In a regression task, we hare given as labels, a real number $\mathcal{Y} = \mathbb{R}$: The task of the learner is then to return the label, clearly we can't expect the learner to return the exact label so we measure the success by some measure of distance: A common loss loss function $\ell(\mathbf{x}, y) = (f(\mathbf{x}) - y)^2$.

**Definition 7.1.** *A general learning problem will consist of a concept class $\mathcal{C}$, a domain $\mathcal{Z}$ and a loss function $\ell : \mathcal{C} \times \mathcal{Z} \to \mathbb{R}_+$. Given a learning problem, and a distribution $D$ we will denote by $\mathcal{L}_D(h)$ the risk function:*

$$\mathcal{L}_D(h) = \mathop{\mathbb{E}}_{z \sim D} [\ell(h, z)]$$

**Definition 7.2.** *[PAC Learning for general loss functions] A learning problem $(\mathcal{C}, \mathcal{Z}, \ell)$ is learnable if there exists an algorithm $A$ and function $m^A_{\mathcal{C}} : (0,1)^2 \to \mathbb{N}$ with the following property:*

*Assume $S = ((x_1, y_1), \ldots, (x_m, y_m))$ is a sample of IID examples generated by some arbitrary distribution $D$. If $S$ is the input of $A$ and $m > m^A_{\mathcal{C}}(\epsilon, \delta)$ then the algorithm returns a hypothesis $h^A_S \in \mathcal{H}$ such that, with probability $1 - \delta$ (over the choice of the $m$ training examples):*

$$\mathcal{L}(h^A_S) < \min_{h \in \mathcal{C}} \mathcal{L}(h) + \epsilon$$

*The function $m^A_{\mathcal{C}}(\epsilon, \delta)$ is referred to as* the sample complexity *of algorithm $A$.*

**Definition 7.3.** *Given a sample $S = \{z_1, \ldots, z_m\}$ we denote the empirical risk:*

$$\mathcal{L}_S(h) = \frac{1}{m} \sum \ell(h, z_i).$$

*An ERM algorithm for a learning problem is an algorithm, that given a sample $S$ returns $h^A_S \in \mathcal{C}$ such that:*

$$\mathcal{L}_S(h^A_S) \leq \min_{h \in \mathcal{C}} \mathcal{L}_S(h)$$

Note that for prediction $\mathcal{Z} = \chi \times \mathcal{Y}$ where the loss function measures the prediction of $h$ as a function of $\mathbf{x}$ w.r.t to $\mathcal{Y}$: However our general notion captures other involved settings such as *unsupervised learning* – these are settings, where the learner doesn't necessarily need to predict the label.

### 7.1.1 The Fund. Theorem Breaks

The first natural question when considering the more general notion of learnability, is whether the Fund. Thm. carries on. Here are a few points to consider:

**VC** The notion of VC dimension is restricted to binary classification: It is not always natural to come up with a natural analogue for more general settings (e.g. what is the extension of VC to 3-labelings? what is the extension of VC in regression?)

**Learnability vs. ERM Learnability.** As a second issue, we will now see an example for a learning problem, where an ERM algorithm will fail to learn (which in turn implies that there is no uniform convergence).

**Example 7.1** (A learning problem with no uniform convergence.). *Consider a learning problem $(\mathcal{C}, \mathcal{Z}, \ell)$ where $\mathcal{Z} = \mathbb{R}$, $\ell(h, z) = h(z)$ and we let*

$$\mathcal{C} = \{f : \mathcal{Z} \to \mathbb{R} : f(z) \geq 0,\ \forall z\},$$

*i.e. $\mathcal{C}$ is the concept class of all positive functions.*

Consider an algorithm that always returns $h_S^A = 0$. Then clearly we have $\mathcal{L}_D(h) = 0$ for every $D$. Since $\mathcal{L}_D(h) \geq 0$ we have that

$$\mathcal{L}(h_S^A) \leq \min_{h \in \mathcal{C}} \mathcal{L}(h)$$

Hence the problem is learnable (with sample complexity 0!).

On the other hand, consider a uniform distribution over $[0, 1]$ and assume a learner that returns the following hypothesis

$$h_S^A(x) \begin{cases} 0 & x \in S \\ 1 & \text{else} \end{cases}$$

Then clearly we have $\mathcal{L}_S(h_S^A) = 0$ but $\mathcal{L}(h_S^A) = 1$.

**Definition 7.4** (Uniform Convergence Property). *We say that a learning problem $(\mathcal{C}, \mathcal{Z}, \ell)$ has the* uniform convergence property *if there exists a function $m : (0, 1)^2 \to \mathbb{N}$ such that for every $\epsilon, \delta \in (0, 1)$ and for every probability distribution $D$ over $\chi$, if $S = ((x_1, y_1), \ldots, (x_m, y_m))$ is a sample of size $m \geq m(\epsilon, \delta)$ drawn IID according to $D$ then with probability at least $(1 - \delta)$ we have that*

$$\sup_{h \in \mathcal{C}} |\mathcal{L}_S(h) - \mathcal{L}(h)| < \epsilon$$

The following parts of the Fund. Thm. remain true and we leave them as an exercise

**Theorem 7.5.** *Let $(\mathcal{C}, \mathcal{Z}, \ell)$ be a learning problem then if $(\mathcal{C}, \mathcal{Z}, \ell)$ has the uniform convergence then it is learnable by any ERM algorithm and the problem is learnable.*

## 7.2 Computational Learning Theory

So far we have restricted our attention to statistical results in learning theory. Namely, we considered an algorithm that receives examples and returns an hypothesis: Even if the hypothesis were restricted to be "efficiently computable functions", the learning algorithm was allowed to be inefficient. We will now add to our model that restriction of efficiency:

**Definition 7.6.** *Given a function $f : (0,1)^2 \to \mathbb{N}$ and a learning task $(\mathcal{C}, \mathcal{Z}, \ell)$, and a learning algorithm $A$. We say that $A$ solves the learning task in time $O(f)$ if there are some constants $c, b$ such that for every probability distribution $D$ and input $(\epsilon, \delta)$ when $A$ has access to samples generated i.i.d by $D$*

- *$A$ terminates after performin at most $cf(\epsilon, \delta)$ operations.*
- *The output $h_S^A$ can be applied on every $z \in \mathcal{Z}$ by performing at most $cf(\epsilon, \delta)$ operations.*
- *$A$ PAC learns the problem $(\mathcal{C}, \mathcal{Z}, \ell)$. i.e. $\mathcal{L}_D(h_S^A) \leq \min_{h \in \mathcal{C}} \mathcal{L}(h)$.*

*Consider a sequence of learning problem $(\mathcal{C}_n, \mathcal{Z}_n, \ell_n)_{n=1}^{\infty}$, parametrized by $n \in \mathbb{N}$. Let $A$ be a learning algorithm designed to solve a problem of this form. Given a function $g : \mathbb{N}(0,1)^2 \to \mathbb{N}$ we say that the runtime of $A$ with respect to the above sequence is $O(g)$ if for all $n$, $A$ solves the problem $(\mathcal{C}_n, \mathcal{Z}_n, \ell_n)$ in time $O(f_n)$ where $f_n = g(n, \cdot, \cdot)$.*

*$A$ is said to be efficient if its runtime is* $\mathrm{poly}(n, 1/\epsilon, 1/\delta)$.

**Remark 1.** *It is straightforward that if a class is not learnable, then it is not efficiently learnable: as we will need to observe unbounded amount of examples (where observing an example, is considered an operation)*

**Example 7.2** (Finite Classes)**.** *Given a finite class we've seen that it can be learnt using an ERM rule with sample complexity $O(\frac{1}{\epsilon^2} \log |H|)$. What is the worst case bound we can give to an ERM rule?*

*To implement an ERM rule we need to verify all hypotheses in $\mathcal{C}_n$ which can be order of $|\mathcal{C}_n|$, therefore a learning problem will be efficiently learnable if $|\mathcal{C}_n| = \mathrm{poly}(n)$.*

## 7.3 Efficient Learning of Half–Spaces, Realizable setting

In this section we discuss efficient algorithms to learn halfspaces over the hypercube. we will for simplicity discuss halfspaces without bias i.e.

$$\mathcal{C}_n = \{f_{\mathbf{w}} : f_{\mathbf{w}}(\mathbf{x}) = \mathrm{sgn}(\mathbf{w} \cdot \mathbf{x}), \mathbf{x} \in \{-1, 1\}^n\}$$

. We've already seen that the VC dimension of halfspaces is $n + 1$. Therefore, if we implement an ERM algorithm over $O(\frac{n+1}{\epsilon} \log 1/\delta)$ examples, we can obtain an efficient algorithm to learn halfspaces.

The idea is to implement an ERM through Linear Program (LP). An LP is a problem of the form

$$\begin{gathered}\text{minimize } \mathbf{w} \cdot \mathbf{u} \\ \text{subject to } A\mathbf{w} \geq \mathbf{v}\end{gathered}$$

For some parameters $\mathbf{u}, \mathbf{v}$ and matrix $A$. Efficient algorithm to solve LP are known to exist (efficient in size of $A$ and $\mathbf{u}$. Therefore we only need to show that we can formulate the ERM as an LP. To see that we can, first note that if there is a solution, then by pertubation we can assume there is $\mathbf{w}$ such that for all $\mathbf{x}_i \in S$ we have $y_i \mathbf{w} \cdot \mathbf{x}_i > 0$ (i.e. strict inequality). By setting $\gamma = \min y_i \mathbf{w} \cdot \mathbf{x}_i$ we and setting $\hat{\mathbf{w}} = \frac{\mathbf{w}}{\gamma}$ we can see that there exists a solution $y_i \hat{\mathbf{w}} \mathbf{x}_i > 1$.

We infact, don't even need $\mathbf{u}$ and we only need to check the feasibility of the problem. Hence, we can now set $\mathbf{u} = 0$ (or any other vector) and set the matrix $A$ such that the $i$-th row $A_i = y_i \mathbf{x}_i^\top$, and $\mathbf{v} = 1$. By the last paragraph, there is a feasible solution – therefore we can find it using an efficient LP algorithm. Any solution to the above program will satisfy $\mathrm{sgn}(\mathbf{w} \cdot \mathbf{x}_i) = y_i$.